Analog models of computations & Effective Church Turing Thesis: Efficient simulation of Turing machines by the General Purpose Analog Computer

Olivier Bournez[1], Daniel S. Graça[2,3], and Amaury Pouly[4]

[1] Ecole Polytechnique, LIX, 91128 Palaiseau Cedex, France.
Olivier.Bournez@lix.polytechnique.fr
[2] CEDMES/FCT, Universidade do Algarve, C. Gambelas, 8005-139 Faro, Portugal. dgraca@ualg.pt
[3] SQIG /Instituto de Telecomunicações, Lisbon, Portugal.
[4] Ecole Normale Supérieure de Lyon, France. Amaury.Pouly@ens-lyon.fr

Abstract. *Are analog models of computations more powerful than classical models of computations?* From a series of recent papers, it is now clear that many realistic analog models of computations are provably equivalent to classical digital models of computations from a *computability* point of view. Take, for example, the probably most realistic model of analog computation, the General Purpose Analog Computer (GPAC) model from Claude Shannon, a model for Differential Analyzers, which are analog machines used from 1930s to early 1960s to solve various problems. It is now known that functions computable by Turing machines are provably exactly those that are computable by GPAC.

This paper is about next step: understanding if this equivalence also holds at the *complexity* level. In this paper we show that the realistic models of analog computation – namely the General Purpose Analog Computer (GPAC) – can simulate Turing machines in a computationally efficient manner. More concretely we show that, modulo polynomial reductions, computations of Turing machines can be simulated by GPACs, without the need of using more (space) resources than those used in the original Turing computation.

As we recently proved that functions computable by a GPAC in a polynomial time with similar restrictions can always be simulated by a Turing machine, this opens the way to, first, a proof that realistic analog models of computations do satisfy the effective Church Turing thesis, and second to a well founded theory of complexity for analog models of computations.

1 Introduction

The Church-Turing thesis is a cornerstone result in theoretical computer science. It states that any (discrete time, digital) computational model which captures the notion of algorithm is computationally equivalent to the Turing machine (see e.g. [19], [23]). It also relates various aspects of models in a very surprising and strong way.

When considering non-discrete time or non-digital models, the situation is far from being so clear. In particular, when considering models working over real numbers, several models are clearly not equivalent [9].

However, a question of interest is whether physically *realistic* models of computation over the real numbers are equivalent, or can be related. Some of the results of non-equivalence involve models, like the BSS model [5], [4], which are claimed not to be physically realistic [9] (although they certainly are interesting from an algebraic perspective), or models which depend critically on the use of exact precision computation for obtaining super-Turing power, e.g. [1], [3].

Realistic models of computation over the reals clearly include the *General Purpose Analog Computer (GPAC)*, an analog continuous-time model of computation and *Computable Analysis*. The GPAC is a mathematical model introduced by Shannon [21] of an earlier analog computer, the Differential Analyzer. The first general-purpose Differential Analyzer is generally attributed to Vannevar Bush [10]. Differential Analyzers have been used intensively up to the 1950's as computational machines to solve various problems from ballistic to aircraft design, before the era of the digital computer [18].

Computable analysis, based on Turing machines, can be considered as today's most used model for talking about computability and complexity over reals. In this approach, real numbers are encoded as sequences of discrete quantities and a discrete model is used to compute over these sequences. More details can be found in the books

[20], [17], [24]. As this model is clearly based on classical (digital and discrete time) models like Turing machines, and that such models are admitted to be realistic models of today's computers, one can clearly consider such an approach to deal with a realistic model of computation.

Understanding whether there could exist something similar to a Church-Turing thesis for analog models of computation, or whether analog models of computation could be more powerful than today's classical models of computation motivated us to try to relate GPAC computable functions to functions computable in the sense of recursive analysis.

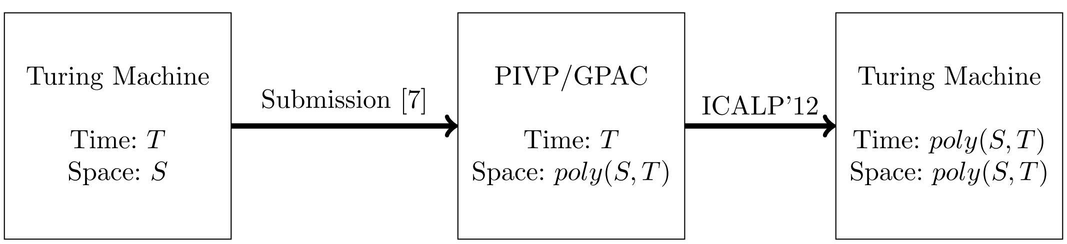
The paper [6] was a first step towards the objective of obtaining a version of the Church-Turing thesis for physically feasible models over the real numbers. This paper proves that, from a computability perspective, Computable Analysis and the GPAC are equivalent: GPAC computable functions are computable and, conversely, functions computable by Turing machines or in the computable analysis sense can be computed by GPACs.

However this is about *computability*, and not *complexity*. This proves that one can not solve more problems using analog models. But this leaves open the intriguing question whether one could solve some problems *faster* using analog models of computations (see e.g. what happens for quantum models of computations...). In other words, the question of whether the above models are equivalent at a computational complexity level remained open. Part of the difficulty stems from finding an appropriate notion of complexity (see e.g. [22], [2]) for analog models of computations.

In the present paper we study both the GPAC and Computable Analysis at a complexity level. In particular, we introduce measures for space complexity and show that, using these measures, both models are equivalent, even at a computational complexity level. Since we already have shown in our previous paper [7] that Turing machines can simulate efficiently GPACs, we will prove in this paper the missing link: GPACs can simulate Turing machines in an efficient manner.

More concretely we show that, modulo polynomial reductions, computations of Turing machines can be simulated by GPACs, without the need of using more (space) resources than those used in the original Turing computation.

In a schematic view, here is the situation that we reach, when relating the constructions presented in this paper with already known results, where PIVP stands for *Polynomial Initial Value Problems*, known to be equivalent to GPACs (see [16] and Section 2.2).



We believe that these results open the way to state that realistic analog models do satisfy the classical Church-Turing thesis in a provable way, both at the computability and complexity level, hence when talking both about Church-Turing thesis and Church-Turing effective thesis.

We believe that this opens the way to a well founded complexity theory for analog models of computations and for continuous dynamical systems.

Notice that it has been observed in several papers that, since continuous time systems might undergo arbitrary space and time contractions, Turing machines, as well as even accelerating Turing machines[5] [14], [13], [12] or even oracle Turing machines, can actually be simulated in an arbitrary short time by ordinary differential equations in an arbitrary short time or space. This is sometimes also called *Zeno's phenomenon*: an infinite number of discrete transitions may happen in a finite time: see e.g. [8].

Such constructions or facts have been deep obstacles to various attempts to build a well founded complexity theory for analog models of computations: see [8] for discussions.

One way to interpret our results is then the following: all these time and space phenomena, or Zeno's phenomena do not hold (or, at least, they do not hold in a problematic manner) for ordinary differential equations corresponding

[5] Similar possibilities of simulating accelerating Turing machines through quantum mechanics are discussed in [11].

to GPACs, that is to say for *realistic* models. This has already been stated at various places. The novelty is first a statement of what "realistic" includes, and second a formal proof of it.

2 GPAC

2.1 Preliminaries

Throughout the paper we will use the following notation: $\|(x_1,\dots,x_n)\|_\infty = \max_{1\leqslant i\leqslant n}|x_i|$ and $\|(x_1,\dots,x_n)\| = \sqrt{|x_1|^2+\cdots+|x_n|^2}$. We will also use the following shortcuts $\pi_i(x_1,\dots,x_k)=x_i$, $\mathrm{int}(x)=\lfloor x\rfloor$, $\mathrm{frac}(x)=x-\lfloor x\rfloor$, $\mathrm{int}_n(x)=\min(n,\mathrm{int}(x))$, $\mathrm{frac}_n(x)=x-\mathrm{int}_n(x)$, and

$$f^{[n]} = \begin{cases}\mathrm{id} & \text{if } n=0\\ f^{[n-1]} & \text{otherwise}\end{cases}$$

In this section, we consider the following ODE

$$\begin{cases}\dot{y} &= p(y)\\ y(t_0) &= y_0\end{cases} \tag{1}$$

where $p:\mathbb{R}^d\to\mathbb{R}^d$ is a vector of polynomials. This is motivated by the fact that this is known [16] that a function is generable by a GPAC iff it is a component of the solution to the initial-value problem (1).

If $p:\mathbb{R}^d\to\mathbb{R}$ a is polynomial, we write $p(X_1,\dots,X_d)=\sum_{|\alpha|\leqslant k}a_\alpha X^\alpha$ where k is degree of p_i that we denote as $\mathrm{d}^\circ p_i$. We also take, as usual, $|\alpha|=\alpha_1+\cdots+\alpha_d$. We also write $\Sigma P=\sum_{|\alpha|\leqslant k}|a_\alpha|$ If $p:\mathbb{R}^d\to\mathbb{R}^d$ is a vector of polynomials, we write $\mathrm{d}^\circ p=\max(\mathrm{d}^\circ p_1,\dots,\mathrm{d}^\circ p_d)$ and $\Sigma p=\max(\Sigma p_1,\dots,\Sigma p_d)$. When it is not ambiguous, for any constant $A\in\mathbb{R}$ we identify A with the constant function $(x_1,\dots,x_k)\mapsto A$.

2.2 Basic properties

It is known [16] that a function is generable by a GPAC iff it is a component of the solution to the initial-value problem (1): formally, a function $f:\mathbb{R}\to\mathbb{R}$ is generable by a GPAC if it belongs to the following class $\mathrm{GPAC}(I)$:

Definition 1. *Let $I\subseteq\mathbb{R}$ be an open interval and $f:I\to\mathbb{R}$. We say that $f\in\mathrm{GPAC}(I)$ if there exists $d\in\mathbb{N}$, a vector of polynomials p, $t_0\in I$ and $y_0\in\mathbb{R}^d$ such that $\forall t\in I, f(t)=y_1(t)$, where $y:I\to\mathbb{R}$ is the unique solution over I of*

$$\begin{cases}\dot{y} &= p(y)\\ y(t_0) &= y_0\end{cases}$$

We want to talk about a subclass of GPAC generable functions, that permits to talk about complexity. This leads to the following definition.

Definition 2. *Let $I\subseteq\mathbb{R}$ be an open interval, $f,g:I\to\mathbb{R}$. We say that $f\in\mathrm{GSPACE}(I,g)$ if there exists $d\in\mathbb{N}$, a vector of polynomials p, $t_0\in I$ and $y_0\in\mathbb{R}^d$ such that $\forall t\in I, f(t)=y_1(t)$ and $\|y(t)\|_\infty\leqslant g(t)$, where $y:I\to\mathbb{R}$ is the unique solution over I of*

$$\begin{cases}\dot{y} &= p(y)\\ y(t_0) &= y_0\end{cases}$$

Let $f:I\to\mathbb{R}^d$. We say that $f\in\mathrm{GSPACE}(I,g)$ if $\forall i,(f_i:I\to\mathbb{R})\in\mathrm{GSPACE}(I,g)$.

The following can be proved (non-trivial missing proofs are in appendix).

Lemma 1. *Let $I,J\subseteq\mathbb{R}$, $f\in\mathrm{GSPACE}(I,s_f)$ and $g\in\mathrm{GSPACE}(J,s_g)$. Then:*

- $f+g, f-g\in\mathrm{GSPACE}(I\cap J, s_f+s_g)$
- $fg\in\mathrm{GSPACE}(I\cap J,\max(s_f,s_g,s_fs_g))$

– $f \circ g \in \text{GSPACE}\,(J, \max(s_g, s_f \circ s_g))$ *if* $g(J) \subseteq I$

Definition 3. *Let* $I \subseteq \mathbb{R}^d$ *be open set and* $f, s_f : I \to \mathbb{R}$*. We say that* $f \in \text{GSPACE}\,(I, s_f)$ *if*

$$\forall J \subseteq \mathbb{R} \text{ open interval }, \forall (g : J \to \mathbb{R}^d) \in \text{GSPACE}\,(J, s_g) \text{ such that } g(J) \subseteq I, \quad f \circ g \in \text{GSPACE}\,(J, \max(s_g, s_f \circ s_g))$$

Remark 1. In the special case of $I \subseteq \mathbb{R}$, Definition 3 matches Definition 2 because of Lemma 1.

Lemma 2. *Let* $I, J \subseteq \mathbb{R}^d$ *be open sets,* $(f : I \to \mathbb{R}^n) \in \text{GSPACE}\,(I, s_f)$ *and* $(g : J \to \mathbb{R}^m) \in \text{GSPACE}\,(J, s_g)$*. Then:*

– $f + g, f - g \in \text{GSPACE}\,(I \cap J, s_f + s_g)$ *if* $n = m$
– $fg \in \text{GSPACE}\,(I \cap J, \max(s_f, s_g, s_f s_g))$ *if* $n = m$
– $f \circ g \in \text{GSPACE}\,(J, \max(s_g, s_f \circ s_g))$ *if* $m = d$ *and* $g(J) \subseteq I$

Definition 4. *Let* $d, e \in \mathbb{N}$ *and* $(i_1, \dots, i_e) \in [\![1, d]\!]^e$ *we define*

$$\pi_d(i_1, \dots, i_e) : \begin{cases} \mathbb{R}^d & \to & \mathbb{R}^e \\ (x_1, \dots, x_d) & \mapsto & (x_{i_1}, \dots, x_{i_e}) \end{cases}$$

Example 1.

$$\pi_4(1, 3) : \begin{cases} \mathbb{R}^4 & \to & \mathbb{R}^2 \\ (x_1, x_2, x_3, x_4) & \mapsto & (x_1, x_3) \end{cases}$$

Lemma 3. *For any* $d, e \in \mathbb{N}$ *such that* $e \leqslant d$*, for any* $(i_1, \dots, i_e) \in [\![1, d]\!]^e$ *and for any* $I \subseteq \mathbb{R}^d$*,*

$$\pi_d(i_1, \dots, i_e) \in \text{GSPACE}\,(I, 0)$$

Lemma 4. *Let* $I \subseteq \mathbb{R}^c$ *and* $J \subseteq \mathbb{R}^e$ *be open sets,* $(f : I \to \mathbb{R}^d) \in \text{GSPACE}\,(I, s_f)$ *and* $p : J \to I$ *a vector of polynomials. Then* $f \circ p \in \text{GSPACE}\,(J, s_f \circ p)$*.*

Lemma 5. $\sin, \tanh \in \text{GSPACE}\,(\mathbb{R}, 1)$

2.3 Main result

The main result of this paper is the following ($\text{poly}(S(e) + T)$ stands for polynomial in $S(e) + T$)):

Theorem 1. *Let* M *be a Turing Machine. Then there exists a vector of polynomials* p *such that, for any input* e *and time* T*, the solution* y *of* (1) *with initial condition* $y(0) =< \phi(e), \psi(S(e), T), ... >$ *(*ϕ *and* ψ *define a simple encoding scheme), where* $S(e)$ *is the space used by* M *on input* e*, satisfies the following properties:*

– *for any integer time* $t \leqslant T$*,* $y(t)$ *fully and unambiguously describes the state of* M *on input* e *at step* t
– *for any* $0 \leqslant t \leqslant T$*,* $\|y(t)\|_\infty \leqslant \text{poly}(S(e) + T)$

3 Turing Machines Simulations

In this section we explain how to simulate a Turing Machine with a GPAC. We would like to simulate a Turing Machine with a polynomially bounded GPAC. As a matter of comparison, it is already known how to simulation any Turing Machine for an arbitrary number of steps using an exponentially bounded GPAC [15].

Our simulations are different from the already known ones in several ways:

– The simulation will only be valid for a certain number of steps: this will be sufficient as we want to talk about (time) complexity, and hence we mostly have a bound on the time of computation.
– The values of the components of the system will be polynomially bounded;

3.1 Helper functions

Our simulation will be performed on a real domain and may be subjected to (small) errors. Thus, to simulate a Turing machine over a large number os steps, we need tools which allow us to keep errors under control. In this section we present functions which are specially designed to fulfill this objective. We call these function *helper functions*. Notice that since functions generated by GPACs are analytic, all helper functions are required to be analytic. As a building block for creating more complex functions, it will be useful to obtain analytic approximations of the functions $\operatorname{int}(x)$ and $\operatorname{frac}(x)$. Notice that we are only concerned about nonnegative numbers so there is no need to discuss the definition of these functions on negative numbers. A graphical representation of the various helper functions we will introduce in this section can be found on Figure 1, Figure 3 and Figure 4.

Definition 5. *For any $x, y, \lambda \in \mathbb{R}$ define $\xi(x, y, \lambda) = \tanh(xy\lambda)$*

The following can be proved (see appendix).

Lemma 6. *For any $x \in \mathbb{R}$ and $\lambda > 0, y \geqslant 1$,*

$$|\operatorname{sgn}(x) - \xi(x, y, \lambda)| < 1/2$$

Furthermore if $|x| \geqslant \lambda^{-1}$ then

$$|\operatorname{sgn}(x) - \xi(x, y, \lambda)| < e^{-y}$$

and $\xi \in \mathrm{GSPACE}\left(\mathbb{R}^3, 1\right)$.

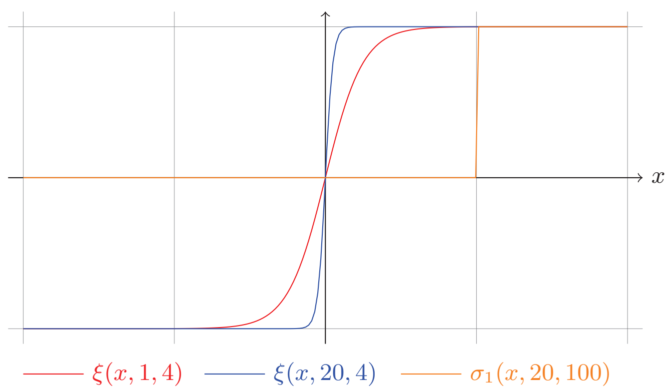


Fig. 1. Graphical representation of ξ and σ_1

Definition 6. *For any $x, y, \lambda \in \mathbb{R}$, define $\sigma_1(x, y, \lambda) = \frac{1+\xi(x-1,y,\lambda)}{2}$*

Corollary 1. *For any $x \in \mathbb{R}$ and $y > 0, \lambda > 2$,*

$$|\operatorname{int}_1(x) - \sigma_1(x, y, \lambda)| \leqslant 1/2$$

Furthermore if $|1 - x| \geqslant \lambda^{-1}$ then

$$|\operatorname{int}_1(x) - \sigma_1(x, y, \lambda)| < e^{-y}$$

and $\sigma_1 \in \mathrm{GSPACE}\left(\mathbb{R}^3, 1\right)$.

Definition 7. *For any $p \in \mathbb{N}$, $x, y, \lambda \in \mathbb{R}$, define $\sigma_p(x, y, \lambda) = \sum_{i=0}^{k-1} \sigma_1(x - i, y + \ln p, \lambda)$*

Lemma 7. *For any $p \in \mathbb{N}$, $x \in \mathbb{R}$ and $y > 0, \lambda > 2$,*

$$|\operatorname{int}_p(x) - \sigma_p(x, y, \lambda)| \leqslant 1/2 + e^{-y}$$

Furthermore if $x < 1 - \lambda^{-1}$ or $x > p + \lambda^{-1}$ or $d(x, \mathbb{N}) > \lambda^{-1}$ then

$$|\operatorname{int}_p(x) - \sigma_p(x, y, \lambda)| < e^{-y}$$

and $\sigma_p \in \mathrm{GSPACE}\left(\mathbb{R}^3, p\right)$.

Finally, we build a square wave like function which we be useful later on.

Definition 8. *For any $t \in \mathbb{R}$, and $\lambda > 0$, define $\theta(t,\lambda) = e^{-\lambda(1-\sin(2\pi t))^2}$*

Lemma 8. *For any $\lambda > 0$, $\rho(\cdot,\lambda)$ is a positive and 1-periodic function bounded by 1, furthermore*

$$\forall t \in [1/2,1], |\theta(t,\lambda)| \leqslant e^{-\lambda}$$

$$\int_0^{\frac{1}{2}} \theta(t,\lambda)dt \geqslant \frac{(e\lambda)^{-\frac{1}{4}}}{\pi}$$

and $\theta \in \mathrm{GSPACE}\left(\mathbb{R}\times\mathbb{R}_+^*, (t,\lambda) \mapsto \max(1,\lambda)\right)$.

Polynomial interpolation In order to implement the transition function of the Turing Machine, we will use polynomial interpolation (Lagrange interpolation). But since our simulation may have to deal with some amount of error in inputs, we have to investigate how this error propagates through the interpolating polynomial.

Lemma 9. *Let $n \in \mathbb{N}$, $x, y \in \mathbb{R}^n$, $K > 0$ be such that $\|x\|_\infty, \|y\|_\infty \leqslant K$, then*

$$\left|\prod_{i=1}^n x_i - \prod_{i=1}^n y_i\right| \leqslant K^{n-1}\sum_{i=1}^n |x_i - y_i|$$

Definition 9 (Lagrange polynomial). *Let $d \in \mathbb{N}$ and $f : G \to \mathbb{R}$ where G is a finite subset of $\mathbb{R}^d$, we define*

$$L_f(x) = \sum_{\bar{x}\in G} f(\bar{x}) \prod_{i=1}^d \prod_{\substack{y\in G\\ y\neq\bar{x}}} \frac{x_i - y_i}{\bar{x}_i - y_i}$$

We recall that by definition, for all $x \in G$, $L_f(x) = f(x)$ so the interesting part is to know what happen for values of x not in G but close to G, that is to relate $L_f(x) - L_f(\tilde{x})$ with $x - \tilde{x}$.

Lemma 10. *Let $d \in \mathbb{N}$, $K > 0$ and $f : G \to \mathbb{R}$, where G is a finite subset of $\mathbb{R}^d$. Then*

$$\forall x, z \in [-K,K]^d, |L_f(x) - L_f(z)| \leqslant A\|x - z\|_\infty \qquad \text{and} \qquad L_f \in \mathrm{GSPACE}\left([-K,K]^d, B\right)$$

where

$$\delta = \min_{x\neq x'\in G} \min_{i=1}^d |x_i - x'_i| \qquad F = \max_{x\in G} |f(x)| \qquad M = K + \max_{x\in G} \|x\|_\infty$$

$$A = |G|F\left(\frac{M}{\delta}\right)^{d(|G|-1)-1} d(|G|-1) \qquad B = |G|F\left(\frac{M}{\delta}\right)^{d(|G|-1)}$$

3.2 Turing Machine

Assumptions Let $\mathcal{M} = (Q, \Sigma, b, \delta, q_0, F)$ be a Turing Machine which will be fixed for the whole simulation. Without loss of generality we assume that:

- When the machine reaches a final state, it stays in this state: so F is useless in what follows.
- $Q = [\![0, m-1]\!]$
- $\Sigma = [\![0, k-2]\!]$ and $b = 0$
- $\delta : Q \times \Sigma \to Q \times \Sigma \times \{L, R\}$, and we identify $\{L, R\}$ with $\{0, 1\}$ ($L = 0$ and $R = 1$).

Consider a configuration $c = (x, \sigma, y)$ of the machine as described in Figure 2. We could encode it as a triple of integers as done in [15] (e.g. if $x_0, x_1, \ldots$ are the digits of x in base k, encode x as the number $x_0 + x_1k + x_2k^2 + \cdots + x_nk^n$), but this encoding is not suitable for our needs. We instead define the *rational encoding* $[c]$ of c as follows.

Definition 10. *Let* $c = (x, s, y, q)$ *be a configuration of* $\mathcal{M}$*, we define the* rational encoding $[c]$ *of* c *as* $[c] = (0.x, s, 0.y, q)$ *where:*

$$0.x = x_0k^{-1} + x_1k^{-2} + \cdots + x_nk^{-n-1} \in \mathbb{Q} \qquad \textit{if} \qquad x = x_0 + x_1k + \cdots + x_nk^n \in \mathbb{N}.$$

The following lemma explains the consequences on the rational encoding of configurations of the assumptions we made for $\mathcal{M}$.

Lemma 11. *Let* c *be a reachable configuration of* $\mathcal{M}$ *and* $[c] = (0.x, \sigma, 0.y, q)$*, then* $0.x \in [0, \frac{k-1}{k}]$ *and similarly for* y*.*

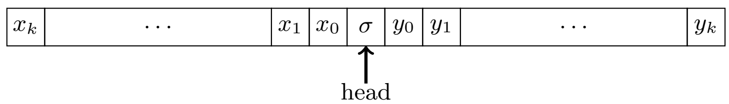


Fig. 2. Turing Machine configuration

Simulation by iteration The first step towards a simulation of a Turing Machine $\mathcal{M}$ using a GPAC is to simulate the transition function of $\mathcal{M}$ with a GPAC-computable function $\text{step}_{\mathcal{M}}$. The next step is to iterate the function $\text{step}_{\mathcal{M}}$ with a GPAC. Instead of considering configurations c of the machine, we will consider its rational configurations $[c]$ and use the helper functions defined previously. Theoretically, because $[c]$ is rational, we just need that the simulation works over rationals. But, in practice, because errors are allowed on inputs, the function $\text{step}_{\mathcal{M}}$ has to simulate the transition function of $\mathcal{M}$ in a manner which tolerates small errors on the input. We recall that δ is the transition function of the $\mathcal{M}$ and we write δ_i the i^{th} component of δ.

Definition 11. *We define:*

$$\text{step}_{\mathcal{M}} : \begin{cases} \mathbb{R}^4 & \longrightarrow & \mathbb{R}^4 \\ \begin{pmatrix} x \\ s \\ y \\ q \end{pmatrix} & \mapsto & \begin{pmatrix} \text{choose}\left[\text{frac}(kx), \frac{y + L_{\delta_2}(q,s)}{k}\right] \\ \text{choose}\left[\text{int}(kx), \text{int}(ky)\right] \\ \text{choose}\left[\frac{y + L_{\delta_2}(q,s)}{k}, \text{frac}(ky)\right] \\ L_{\delta_1}(q, s) \end{pmatrix} \end{cases}$$

where

$$\text{choose}[a, b] = (1 - L_{\delta_3}(q, s))a + L_{\delta_3}(q, s)b$$

The function $\text{step}_{\mathcal{M}}$ simulates the transition function of the Turing Machine $\mathcal{M}$, as shown in the following result.

Lemma 12. *Let* $c_0, c_1, \ldots$ *be the sequence of configurations of* $\mathcal{M}$ *starting from* c_0*. Then*

$$\forall n \in \mathbb{N}, [c_n] = \text{step}_{\mathcal{M}}{}^{[n]}([c_0])$$

Now we want to extend the function $\text{step}_{\mathcal{M}}$ to work not only on rationals coding configuration but also on reals close to configurations, in a way which tolerates small errors on the input. That is we want to build a robust approximation of $\text{step}_{\mathcal{M}}$. We already have some results on L thanks to Lemma 10. We also have some results on $\text{int}(\cdot)$ and $\text{frac}(\cdot)$. However, we need to pay attention to the case of nearly empty tapes.

Indeed, consider the case where, say, the tape on the left of the head contains a single character s. Then $x = 0.s = sk^{-1}$. Now assume that the head moves left. After the move we should be in the configuration $(0, s, \_, \_)$. Looking at the definition of $\text{step}_{\mathcal{M}}$, one sees that this works because $\text{int}(kx) = \text{int}(s) = s$. However if one perturbes a little bit x to $x - \epsilon$, then $\text{int}(k(x - \epsilon)) = s - 1$. To overcome this difficulty, we will use to our advantage the choice of k and its main consequence, Lemma 11. Indeed, if we shift kx by a small amount, we can allow a small perturbation. Since integers are the worst case scenario for our approximation of $\text{int}(\cdot)$, we center the unreachable range for the tape on it, thus considering $\text{int}(kx + \frac{1}{2k})$. The same applies to $\text{frac}(kx)$ with a caveat: we will use $kx - \text{int}(kx + \frac{1}{2k})$ instead.

Definition 12. *Define:*

$$\overline{\mathrm{step}}_{\mathcal{M}}(\tau,\lambda):\begin{cases}\mathbb{R}^4 & \longrightarrow & \mathbb{R}^4\\ \begin{pmatrix}x\\s\\y\\q\end{pmatrix} & \mapsto & \begin{pmatrix}\mathrm{choose}\left[\overline{\mathrm{frac}}(kx),\frac{x+L_{\delta_2}(q,s)}{k},q,s\right]\\ \mathrm{choose}\left[\overline{\mathrm{int}}(kx),\overline{\mathrm{int}}(ky),q,s\right]\\ \mathrm{choose}\left[\frac{y+L_{\delta_2}(q,s)}{k},\overline{\mathrm{frac}}(ky),q,s\right]\\ L_{\delta_1}(q,s)\end{pmatrix}\end{cases}$$

where

$$\mathrm{choose}[a,b,q,s]=(1-L_{\delta_3}(q,s))a+L_{\delta_3}(q,s)b$$

$$\overline{\mathrm{int}}(x)=\sigma_k\left(x+\frac{1}{2k},\tau,\lambda\right)$$

$$\overline{\mathrm{frac}}(x)=x-\overline{\mathrm{int}}(x)$$

We now show that $\overline{\mathrm{step}}_{\mathcal{M}}$ is a robust version of $\mathrm{step}_{\mathcal{M}}$. We first begin with a lemma about function *choose*.

Lemma 13. *There exists $A_3>0$ and $B_3>0$ such that $\forall q,\bar{q},s,\bar{s},a,b,\bar{a},\bar{b}\in\mathbb{R}$, if*

$$\left\|(\bar{a},\bar{b})\right\|_\infty\leqslant M\qquad\text{and}\qquad q\in Q,s\in\Sigma\qquad\text{and}\qquad\|(q,s)-(\bar{q},\bar{s})\|_\infty\leqslant 1$$

then

$$\left|\mathrm{choose}[a,b,q,s]-\mathrm{choose}[\bar{a},\bar{b},\bar{q},\bar{s}]\right|\leqslant\left\|(a,b)-(\bar{a},\bar{b})\right\|_\infty+2MA_3\|(q,s)-(\bar{q},\bar{s})\|_\infty$$

Furthermore, $\mathrm{choose}\in\mathrm{GSPACE}\left(\mathbb{R}^2\times[-m,m]\times[-k,k],(a,b,q,s)\mapsto(1+B_3)(a+b)\right)$.

Lemma 14. *There exists $A_1,A_2,A_3,B_1,B_2,B_3>0$ such that for any $\tau,\lambda>0$, any valid rational configuration $c=(x,s,y,q)\in\mathbb{R}^4$ and any $\bar{c}=(\bar{x},\bar{s},\bar{y},\bar{q})\in\mathbb{R}^4$, if*

$$\|(x,y)-(\bar{x},\bar{y})\|_\infty\leqslant\frac{1}{2k^2}-\frac{1}{k\lambda}\qquad\text{and}\qquad\|(q,s)-(\bar{q},\bar{s})\|_\infty\leqslant 1$$

then

$$\begin{array}{ll}\text{for } p\in\{1,3\} & |\mathrm{step}_{\mathcal{M}}(c)_p-\overline{\mathrm{step}}_{\mathcal{M}}(\tau,\lambda)(\bar{c})_p|\leqslant k\|(x,y)-(\bar{x},\bar{y}\|_\infty+(1+2A_3)\left(e^{-\tau}+\frac{A_2}{k}\|(q,s)-(\bar{q},\bar{s})\|_\infty\right)\\ & |\mathrm{step}_{\mathcal{M}}(c)_2-\overline{\mathrm{step}}_{\mathcal{M}}(\tau,\lambda)(\bar{c})_2|\leqslant 2A_3k\|(q,s)-(\bar{q},\bar{s})\|_\infty+e^{-\tau}\\ & |\mathrm{step}_{\mathcal{M}}(c)_4-\overline{\mathrm{step}}_{\mathcal{M}}(\tau,\lambda)(\bar{c})_4|\leqslant A_1\|(q,s)-(\bar{q},\bar{s})\|_\infty\end{array}$$

Furthermore,

$$\overline{\mathrm{step}}_{\mathcal{M}}\in\mathrm{GSPACE}\left((\mathbb{R}_+^*)^2\times[-1,1]\times[-m,m]\times[-1,1]\times[-k,k],B_1+(1+B_3)(2k+1+B_2k^{-1})\right)$$

Proof. To apply Lemma 13, we need two kinds of results: bounds on the difference between the argument of choose and bounds on the argument themselves. Let A_1 and A_2 be the constants coming from Lemma 10 applied to δ_1 and δ_2 (because Q and Σ are finite and thus bounded and $(\bar{q},\bar{s})$ is bounded by hypothesis). To improve readability, we write $\Delta=\|(q,s)-(\bar{q},\bar{s})\|_\infty$.

- We first show that $|\mathrm{int}(kx)-\overline{\mathrm{int}}(k\bar{x})|\leqslant e^{-\tau}$: since c is a valid configuration, $x=0.uz$ where $u\in[\![0,k-2]\!]$ and $z\in[0,\frac{k-1}{k}]$ by Lemma 11. By hypothesis, $|x-\bar{x}|\leqslant\frac{1}{2k^2}-\frac{1}{k\lambda}$ so $|kx-k\bar{x}|\leqslant\frac{1}{2k}-\frac{1}{\lambda}$. Notice that $kx=u.z$ thus $kx\in[u,u+\frac{k-1}{k}]$ and as a consequence $k\bar{x}\in[u-\frac{1}{2k}+\frac{1}{\lambda},u+\frac{k-1}{k}+\frac{1}{2k}-\frac{1}{\lambda}]$. Finally, $k\bar{x}+\frac{1}{2k}\in[u+\frac{1}{\lambda},u+1-\frac{1}{\lambda}]$. By Lemma 7, $|\sigma_k(kx+\frac{1}{2k},\lambda,\tau)-\mathrm{int}_k(kx+\frac{1}{2k})|\leqslant e^{-\tau}$. But $kx+\frac{1}{2k}\in[u+\frac{1}{2k},u+1-\frac{1}{2k}]$ so $\mathrm{int}_k(kx+\frac{1}{2k})=\mathrm{int}(kx)=u$.
- It is then easy to see that $|\mathrm{frac}(kx)-\overline{\mathrm{frac}}(k\bar{x})|\leqslant k|x-\bar{x}|+e^{-\tau}$.
- By Lemma 10, $|L_{\delta_2}(\bar{q},\bar{s})-L_{\delta_2}(q,s)|\leqslant A_2\Delta$.
- Thus, $\left|\frac{x+L_{\delta_2}(q,s)}{k}-\frac{\bar{x}+L_{\delta_2}(\bar{q},\bar{s})}{k}\right|\leqslant\frac{|x-\bar{x}|+A_2\Delta}{k}$
- As a consequence, $|\overline{\mathrm{int}}(k\bar{x})|\leqslant k-2+e^{-\tau}$ since $|\overline{\mathrm{int}}(k\bar{x})|\leqslant|\mathrm{int}(kx)|+|\mathrm{int}(kx)-\overline{\mathrm{int}}(k\bar{x})|$ and $|\mathrm{int}(kx)|\leqslant k-2$

– Similarly, $|\overline{\mathrm{frac}}(k\bar{x})| \leqslant \frac{k-1}{k} + k|x-\bar{x}| + e^{-\tau} \leqslant 1 + e^{-\tau}$ since $|\operatorname{frac}(kx)| \leqslant \frac{k-1}{k}$ by Lemma 11.
– Also, $|L_{\delta_2}(\bar{q},\bar{s})| \leqslant k-2+A_2\varDelta$ since $|L_{\delta_2}(q,s)| \leqslant k-2$.
– Finally, $\left|\frac{\bar{x}+L_{\delta_2}(\bar{q},\bar{s})}{k}\right| \leqslant \frac{|x|+|x-\bar{x}|+k-2+A_2\varDelta}{k} \leqslant \frac{\frac{k-1}{k}+\frac{1}{2k^2}+k-2+A_2\varDelta}{k} \leqslant 1+\frac{A_2}{k}$ since $\varDelta \leqslant 1$.

Now applying Lemma 13 four times gives the result. We write the computation for the two interesting cases:

$$\begin{aligned}
|\operatorname{step}_{\mathcal{M}}(c)_1 - \overline{\operatorname{step}}_{\mathcal{M}}(\tau,\lambda)(\bar{c})_1| &\leqslant \max\left[k|x-\bar{x}|+e^{-\tau}, \frac{|x-\bar{x}|+A_2\varDelta}{k}\right] + 2A_3\max\left[1+e^{-\tau}, 1+\frac{A_2}{k}\right]\varDelta \\
&\leqslant k|x-\bar{x}|+e^{-\tau}+\frac{A_2\varDelta}{k}+2A_3\left(1+e^{-\tau}+\frac{A_2}{k}\right)\varDelta \\
&\leqslant k|x-\bar{x}|+(1+2A_3)\left(e^{-\tau}+\frac{A_2}{k}\varDelta\right)
\end{aligned}$$

$$\begin{aligned}
|\operatorname{step}_{\mathcal{M}}(c)_2 - \overline{\operatorname{step}}_{\mathcal{M}}(\tau,\lambda)(\bar{c})_2| &\leqslant e^{-\tau}+2A_3(k-2+e^{-\tau})\varDelta \\
&\leqslant e^{-\tau}+2A_3k\varDelta
\end{aligned}$$

We summarize the previous lemma into the following simpler form.

Corollary 2. *For any $\tau,\lambda > 0$, any valid rational configuration $c=(x,s,y,q)\in\mathbb{R}^4$ and any $\bar{c}=(\bar{x},\bar{s},\bar{y},\bar{q})\in\mathbb{R}^4$, if*

$$\|(x,y)-(\bar{x},\bar{y})\|_\infty \leqslant \frac{1}{2k^2}-\frac{1}{k\lambda} \qquad \text{and} \qquad \|(q,s)-(\bar{q},\bar{s})\|_\infty \leqslant 1$$

then

$$\left\|\operatorname{step}_{\mathcal{M}}(c)-\overline{\operatorname{step}}_{\mathcal{M}}(\tau,\lambda)(\bar{c})\right\|_\infty \leqslant O(1)(e^{-\tau}+\|c-\bar{c}\|_\infty)$$

Furthermore,

$$\overline{\operatorname{step}}_{\mathcal{M}} \in \operatorname{GSPACE}\left((\mathbb{R}_+^*)^2\times[-1,1]\times[-m,m]\times[-1,1]\times[-k,k], O(1)\right)$$

3.3 Iterating functions with differential equations

We will use a special kind of differential equations to perform the iteration of a map with differential equations. In essence, it relies on the following core differential equation

$$\dot{x}(t) = A\phi(t)(g-x(t)) \tag{Reach}$$

We will see that with proper assumptions, the solution converges very quickly to the *goal* g. However, (Reach) is a simplistic idealization of the system so we need to consider a perturbed equation where the goal is not a constant anymore and the derivative is subject to small errors

$$\dot{x}(t) = A\phi(t)(\bar{g}(t)-x(t))+E(t) \tag{ReachPerturbed}$$

We will again see that, with proper assumptions, the solution converges quickly to the *goal* within a small error. Finally we will see how to build a differential equation which iterates a map within a small error.

We first focus on (Reach) and then (ReachPerturbed) to show that they behave as expected. In this section we assume ϕ is a C^1 function.

Lemma 15. *Let x be a solution of* (Reach)*, let $T,\lambda > 0$ and assume $A \geqslant \frac{\lambda}{\int_0^T \phi(u)du}$ then $|x(T)-g| \leqslant |g-x(0)|e^{-\lambda}$.*

Proof. Check that $x(t) = g+(x(0)-g)e^{-A\int_0^T \phi(u)du}$ is the unique solution of (Reach), which gives the result immediately.

Lemma 16. *Let $T, \lambda > 0$ and let x be the solution of* (ReachPerturbed) *with initial condition $x(0) = x_0$. Assume $|\bar{g}(t) - g| \leqslant \eta$, $A \geqslant \frac{\lambda}{\int_0^T \phi(u)du}$ and $E(t) = 0$ for $t \in [0, T]$. Then*

$$|x(T) - g| \leqslant \eta(1 + e^{-\lambda}) + |x_0 - g|e^{-\lambda}$$

Proof. Let x^+, x^- be the respective solutions of $\dot{x} = A\phi(t)(g \pm \eta - x(t))$ with initial condition $x(0) = x_0$. We will show that $\forall t \in [0, T], x^-(t) \leqslant x(t) \leqslant x^+(t)$.

Consider $f(t, u) = A\phi(t)(\bar{g}(t) - u) + E(t)$ and $f^\pm(t, u) = A\phi(t)(g \pm \eta - u)$. Then x satisfies $\dot{x}(t) = f(t, x(t))$ and $x^\pm$ satisfy $\dot{x}^\pm(t) = f^\pm(t, x(t))$. It is easy to see that for any $t \in [0, T]$ and any $u \in \mathbb{R}$, $f^-(t, u) \leqslant f(t, u) \leqslant f^+(t, u)$. Thus, by a classical result of differential inequations, $x^-(t) \leqslant x(t) \leqslant x^+(t)$ for any $t \in [0, T]$.

We prove that $|x^\pm(T) - g| \leqslant \eta(1 + e^{-\lambda}) + |x_0 - g|e^{-\lambda}$ using Lemma 15 and the fact that $x^-(t) \leqslant x(t) \leqslant x^+(t)$.

We can now define a system that simulates the iteration of a function using a system based on (ReachPerturbed).

Definition 13. *Let $d \in \mathbb{N}$, $F : \mathbb{R}^d \to \mathbb{R}^d$, $\lambda \geqslant 1, \mu \geqslant 0$, we define*

$$\begin{cases} A = 10(\lambda + \mu)^2 \\ B = 4(\lambda + \mu) \\ \dot{z}_i(t) = A\theta(t, B)(F_i(u(t)) - z_i(t)) \\ \dot{u}_i(t) = A\theta(t - 1/2, B)(z_i(t) - u_i(t)) \end{cases} \tag{Iterate}$$

Theorem 2. *Let $d \in \mathbb{N}$, $F : \mathbb{R}^d \to \mathbb{R}^d$, $\lambda \geqslant 1$, $\mu \geqslant 0$, $c_0 \in \mathbb{R}^d$. Assume z, u are solutions to* (Iterate) *and let ΔF and $M \geqslant 1$ be such that*

$$\forall k \in \mathbb{N}, \forall \varepsilon > 0, \forall x \in]-\varepsilon, \varepsilon[^d, \left\| F^{[k+1]}(c_0) - F\left(F^{[k]}(c_0) + x\right) \right\|_\infty \leqslant \Delta F(\varepsilon)$$

$$\forall t \geqslant 0, \|u(t)\|_\infty, \|z(t)\|_\infty, \|F(u(t))\|_\infty \leqslant M = e^\mu$$

and consider

$$\begin{cases} \varepsilon_0 = \|u(0) - c_0\|_\infty \\ \varepsilon_{k+1} = (1 + 3e^{-\lambda})\Delta F(\varepsilon_k + 2e^{-\lambda}) + 5e^{-\lambda} \end{cases}$$

Then

$$\forall k \in \mathbb{N}, \left\| u(k) - F^{[k]}(c_0) \right\|_\infty \leqslant \varepsilon_k$$

Furthermore, if $F \in \mathrm{GSPACE}\left([-M, M]^d, s_F\right)$ then $u \in \mathrm{GSPACE}\left(\left(\mathbb{R}_+^*\right)^3, (\lambda, \mu, t) \mapsto \max(1, 4(\lambda + \mu), s_F(M))\right)$.

Proof. First we show that $AMe^{-B} \leqslant e^{-\lambda}$:

$$\frac{e^{B-\lambda}}{AM} \geqslant \frac{e^{3(\lambda+\mu)}}{10(\lambda + \mu)^2} \geqslant 1 \qquad \text{because } \lambda + \mu \geqslant 1 \text{ by the study of } \frac{e^{3x}}{10x^2}$$

Second we show that $A \geqslant (\lambda + \mu)\pi(eB)^{\frac{1}{4}}$:

$$\frac{A}{(\lambda + \mu)\pi(eB)^{\frac{1}{4}}} = \frac{10(\lambda + \mu)^{\frac{3}{4}}}{\pi(4e)^{1/4}} \geqslant \frac{10}{\pi(4e)^{1/4}} \geqslant 1 \qquad \text{because } \lambda + \mu \geqslant 1$$

We prove the result by induction on k. There is nothing to prove for $k = 0$ since $F^{[0]}(c_0) = c_0$. Now let $k \in \mathbb{N}$ and assume $\left\|u(k) - F^{[k]}(c_0)\right\|_\infty \leqslant \varepsilon_k$. We will work in two steps: first we consider the evolution of u and z on $[k, k + 1/2]$ and then $[k + 1/2, k + 1]$.

- Notice that for any $t \in [k, k + 1/2]$, by Lemma 8, $|\theta(t - 1/2, B)| \leqslant e^{-B}$. Thus $|\dot{u}_i(t)| \leqslant Ae^{-B}2M \leqslant 2e^{-\lambda}$. Hence $|u_i(t) - u_i(k)| \leqslant 2e^{-\lambda}$. By induction hypothesis, $\left\|u(k) - F^{[k]}(c_0)\right\|_\infty \leqslant \varepsilon_k$ thus $\left\|u(t) - F^{[k]}(c_0)\right\|_\infty \leqslant \varepsilon_k + 2e^{-\lambda}$. Finally, by hypothesis, $\left\|F(u(t)) - F^{[k+1]}(c_0)\right\|_\infty \leqslant \Delta F(\varepsilon_k + 2e^{-\lambda})$.

  Now thanks for Lemma 8, $\int_k^{k+1/2} \theta(t, B)dt \geqslant \frac{(eB)^{-\frac{1}{4}}}{\pi} \geqslant \frac{\lambda+\mu}{A}$. Thus the hypothesis of Lemma 16 are met and we have $\left\|z(k + 1/2) - F^{[k+1]}(c_0)\right\|_\infty \leqslant (1 + e^{-\lambda-\mu})\Delta F(\varepsilon_k + 2e^{-\lambda}) + 2Me^{-\lambda-\mu} \leqslant (1 + e^{-\lambda})\Delta F(\varepsilon_k + 2e^{-\lambda}) + 2e^{-\lambda}$.

– Similarly, for any $t \in [k+1/2, k+1]$, by Lemma 8, $\|z(t) - z(k+1/2)\|_\infty \leqslant 2e^{-\lambda}$ thus $\left\|z(t) - F^{[k+1]}(c_0)\right\|_\infty \leqslant (1+e^{-\lambda})\Delta F(\varepsilon_k + 2e^{-\lambda}) + 4e^{-\lambda}$.
Now apply Lemma 16 as before to u and we get that

$$\begin{aligned}\left\|u(k+1) - F^{[k+1]}(c_0)\right\|_\infty &\leqslant (1+e^{-\lambda})\left((1+e^{-\lambda})\Delta F(\varepsilon_k + 2e^{-\lambda}) + 2e^{-\lambda}\right) + 2e^{-\lambda} \\ &\leqslant (1+3e^{-\lambda})\Delta F(\varepsilon_k + 2e^{-\lambda}) + 5e^{-\lambda}\end{aligned}$$

3.4 Turing Machine simulation with differential equations

In this section, we will use results of both Section 3.2 and Section 3.3 to simulate Turing Machines with differential equations. Indeed, in Section 3.2 we showed that we could simulate a Turing Machine by iterating a robust real map, and in Section 3.3 we showed how to efficiently iterate a robust map with differential equations. Now we just have to connect these results

Lemma 17. *Let $a > 1$ and $b \geqslant 0$, assume $u \in \mathbb{R}^{\mathbb{N}}$ satisfies $u_{n+1} \leqslant au_n + b, forn \geqslant 0$. Then*

$$u_n \leqslant a^n u_0 + b\frac{a^n - 1}{a-1}, \quad n \geqslant 0$$

Theorem 3. *Let $\mathcal{M}$ be a Turing Machine as in Section 3.2, then there is $f_{\mathcal{M}} \in \mathrm{GSPACE}\left(\left(\mathbb{R}_+^*\right)^3, s_f\right)$ such that for any sequence $c_0, c_1, \ldots,$ of configurations of $\mathcal{M}$ starting from the initial configuration c_0,*

$$\forall S, T \in \mathbb{R}_+^*, \forall n \leqslant T, \|[c_n] - f_{\mathcal{M}}(S,T,n)\|_\infty \leqslant e^{-S}$$

and

$$\forall S, T \in \mathbb{R}_+^*, \forall n \leqslant T, s_u(S,T,n) = O(poly(S,T))$$

Proof. Let $\tau > 0$ (to be fixed later) and apply Theorem 2 to $F = \overline{\mathrm{step}}_{\mathcal{M}}(\tau, 4k)$. By Corollary 2, $\exists K_1, K_2$ such that

$$\Delta F(\varepsilon) = K_1(e^{-\tau} + \varepsilon)$$

and

$$\forall x \in \Lambda = [-1,1] \times [-m,m] \times [-1,1] \times [-k,k], \|F(x)\|_\infty \leqslant K_2$$

Let $M = K_2 + 1$. The recurrence relation of ε

$$\begin{cases}\varepsilon_0 = \|u(0) - c_0\|_\infty \\ \varepsilon_{k+1} = (1+3e^{-\lambda})\Delta F(\varepsilon_k + 2e^{-\lambda}) + 5e^{-\lambda}\end{cases}$$

now simplifies to (using that $e^{-\lambda} \leqslant 1$)

$$\begin{aligned}\varepsilon_{k+1} &\leqslant (1+3e^{-\lambda})K_1(e^{-\tau} + \varepsilon_k + 2e^{-\lambda}) + 5e^{-\lambda} \\ &\leqslant K_1(1+3e^{-\lambda})\varepsilon_k + 2K_1(1+3e^{-\lambda})e^{-\lambda} + 5e^{-\lambda} \\ &\leqslant \underbrace{K_1(1+3e^{-\lambda})}_{a}\varepsilon_k + \underbrace{(8K_1+5)e^{-\lambda}}_{b}\end{aligned}$$

Now apply Lemma 17 to get an explicit expression

$$\varepsilon_n \leqslant a^n u_0 + b\frac{a^n - 1}{a-1}$$

If we take as initial condition the exact rational configuration $[c_0]$, we immediately get that $u_0 = 0$. Let $K_3 = 4K_1$, then $a \leqslant K_3$. Pick $\lambda = S + T\log(K_3) + \log(8K_1 + 5)$. Then $\varepsilon_T \leqslant e^{-S}$.
We check with Theorem 2 that $\|u(t)\|_\infty, \|z(t)\|_\infty \leqslant M$ for $t \leqslant T$ since $\varepsilon_T \leqslant 1$.

Finally, $u \in \mathrm{GSPACE}\left(\left(\mathbb{R}_+^*\right)^3, \underbrace{(\lambda,\mu,t) \mapsto \max(1, 4(\lambda+\mu), s_F(M))}_{s_u}\right)$ and $s_u = O(poly(S,T))$.

4 Acknowledgments

D.S. Graça was partially supported by *Fundação para a Ciência e a Tecnologia* and EU FEDER POCTI/POCI via SQIG - Instituto de Telecomunicações through the FCT project PEst-OE/EEI/LA0008/2011.

Olivier Bournez and Amaury Pouly were partially supported by ANR project SHAMAN, by DGA, and by DIM LSC DISCOVER project.

References

1. E. Asarin and O. Maler. Achilles and the tortoise climbing up the arithmetical hierarchy. *J. Comput. System Sci.*, 57(3):389–398, 1998.
2. A. Ben-Hur, H. T. Siegelmann, and S. Fishman. A theory of complexity for continuous time systems. *J. Complexity*, 18(1):51–86, 2002.
3. V. D. Blondel, O. Bournez, P. Koiran, and J. N. Tsitsiklis. The stability of saturated linear dynamical systems is undecidable. *J. Comput. System Sci.*, 62:442–462, 2001.
4. L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation.* Springer, 1998.
5. L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bull. Amer. Math. Soc.*, 21(1):1–46, 1989.
6. O. Bournez, M. L. Campagnolo, D. S. Graça, and E. Hainry. Polynomial differential equations compute all real computable functions on computable compact intervals. *J. Complexity*, 23(3):317–335, 2007.
7. O. Bournez, D. S. Graça, and A. Pouly. On the complexity of solving initial value problems. Submitted to the conference ISSAC 2012: International Symposium on Symbolic and Algebraic Computation.
8. Olivier Bournez and Manuel L. Campagnolo. *New Computational Paradigms. Changing Conceptions of What is Computable*, chapter A Survey on Continuous Time Computations, pages 383–423. Springer-Verlag, New York, 2008.
9. V. Brattka. The emperor's new recursiveness: the epigraph of the exponential function in two models of computability. In M. Ito and T. Imaoka, editors, *Words, Languages & Combinatorics III*, Kyoto, Japan, 2000. ICWLC 2000.
10. V. Bush. The differential analyzer. A new machine for solving differential equations. *J. Franklin Inst.*, 212:447–488, 1931.
11. C. S. Calude and B. Pavlov. Coins, quantum measurements, and Turing's barrier. *Quantum Information Processing*, 1(1-2):107–127, April 2002.
12. B. Jack Copeland. Accelerating Turing machines. *Minds and Machines*, 12:281–301, 2002.
13. J. Copeland. Even Turing machines can compute uncomputable functions. In J. Casti, C. Calude, and M. Dinneen, editors, *Unconventional Models of Computation (UMC'98)*, pages 150–164. Springer, 1998.
14. E. B. Davies. Building infinite machines. *The British Journal for the Philosophy of Science*, 52:671–682, 2001.
15. D. S. Graça, M. L. Campagnolo, and J. Buescu. Computability with polynomial differential equations. *Adv. Appl. Math.*, 40(3):330–349, 2008.
16. D. S. Graça and J. F. Costa. Analog computers and recursive functions over the reals. *J. Complexity*, 19(5):644–664, 2003.
17. K.-I Ko. *Computational Complexity of Real Functions.* Birkhäuser, 1991.
18. J. M. Nyce. Guest editor's introduction. *IEEE Ann. Hist. Comput.*, 18:3–4, 1996.
19. P. Odifreddi. *Classical Recursion Theory*, volume 1. Elsevier, 1989.
20. M. B. Pour-El and J. I. Richards. *Computability in Analysis and Physics.* Springer, 1989.
21. C. E. Shannon. Mathematical theory of the differential analyzer. *J. Math. Phys. MIT*, 20:337–354, 1941.
22. H. T. Siegelmann, A. Ben-Hur, and S. Fishman. Computational complexity for continuous time dynamics. *Phys. Rev. Lett.*, 83(7):1463–1466, 1999.
23. M. Sipser. *Introduction to the Theory of Computation.* Course Technology, 2nd edition, 2005.
24. K. Weihrauch. *Computable Analysis: an Introduction.* Springer, 2000.

A Proof of Lemma 1

Proof. We first consider the case of $f \pm g$ and fg. Without loss of generality we can assume that $\exists t_0 \in K = I \cap J$. Then f and g can be taken as components of the solution of two PIVP with the initial conditions sharing the same time t_0

$$\begin{cases} \dot{y} &= p(y) \\ y(t_0) &= y_0 \end{cases} \qquad \begin{cases} \dot{z} &= q(z) \\ z(t_0) &= z_0 \end{cases} \qquad \begin{cases} f(t) &= y_1(t) \\ g(t) &= z_1(t) \end{cases}$$

Then $f \pm g$ and fg are a components of the solution of the following two PIVPs

$$\begin{cases} \dot{y} &= p(y) \\ y(t_0) &= y_0 \\ \dot{z} &= p(z) \\ z(t_0) &= z_0 \\ \dot{u} &= p_1(y) \pm q_1(z) \\ u(t_0) &= y_{01} \pm z_{01} \end{cases} \qquad \begin{cases} \dot{y} &= p(y) \\ y(t_0) &= y_0 \\ \dot{z} &= p(z) \\ z(t_0) &= z_0 \\ \dot{u} &= p_1(y)z_1 + y_1 q_1(z) \\ u(t_0) &= y_{01} z_{01} \end{cases}$$

It follows pretty obviously that $s_f + s_g$ and $\max(s_f, s_g, s_f s_g)$ are natural bounds on the variables of the system.

Now consider the case of $f \circ g$: assume that there are t_0, z_0 such that f and g can be taken as components of the solution of two PIVP with the initial conditions $t_1 = g(t_0)$ for the second one

$$\begin{cases} \dot{y} &= p(y) \\ y(t_1) &= f(t_1) \end{cases} \qquad \begin{cases} \dot{z} &= q(z) \\ z(t_0) &= z_0 \end{cases} \qquad \begin{cases} f(t) &= y_1(t) \\ g(t) &= z_1(t) \end{cases}$$

Now consider $u(t) = y(g(t))$. It is easy to see that $\dot{u}(t) = \dot{g}(t)\dot{y}(g(t))$, so one can conclude that $f \circ g$ is a last component of the solution to the system

$$\begin{cases} \dot{z} &= q(z) \\ z(t_0) &= z_0 \\ \dot{u} &= p(u) q_1(z) \\ u(t_0) &= f(z_0) \end{cases}$$

The bound on the components of the system follows directly.

B Proof of Lemma 3

Proof. Let J be an open interval, let $(g : J \to \mathbb{R}^d) \in \text{GSPACE}\,(J, s_g)$ such that $g(J) \subseteq I$. Then $\pi_d(i_1, \ldots, i_e) \circ g = (g_{i_1}, \ldots, g_{i_e}) \in \text{GSPACE}\,(I, s_g)$. This concludes the proof since $s_g = \max(s_g, 0 \circ s_g)$.

C Proof of Lemma 6

Proof. Since $\text{sgn}(\cdot)$ and $\xi(\cdot, y, \lambda)$ are odd functions, we can assume that $x \geqslant \lambda^{-1}$. A simple calculus shows that $0 \leqslant 1 - \tanh(x) \leqslant e^{-t}$ for $t \geqslant 1$, thus if $x \geqslant \lambda^{-1}$, $xy\lambda \geqslant y \geqslant 1$ which gives the result.

The last result is a direct consequence of Lemma 4 and Lemma 5.

D Proof of Corollary 1

Proof. The first part of the result is a direct consequence of Lemma 6 and the fact that $\text{int}_1(x) = \text{sgn}(x-1)/2 + 1/2$.

Let $p(X) = \frac{1+X}{2}$ and $q(X, Y, Z) = (X - 1, Y, Z)$ Then $p \in \text{GSPACE}\,(\mathbb{R}, p)$. Furthermore, $\sigma_1 = p \circ \xi \circ q$. Thus by Lemma 4, $\xi \circ q \in \text{GSPACE}\left(\mathbb{R}^3, 1\right)$ and by Lemma 2, $\sigma_1 \in \text{GSPACE}\left(\mathbb{R}^3, \max(1, p \circ 1)\right)$. The results follows since $p \circ 1 = 1$.

E Proof of Lemma 7

Proof. Let $x \in \mathbb{R}$. In all cases we apply Corollary 1 multiples times.

- If $x < 1 - \lambda^{-1}$ then for all $0 \leqslant i \leqslant p - 1$, $|\sigma_1(x - i, y + \ln p, \lambda)| \leqslant e^{-y - \ln p} = \frac{e^{-y}}{p}$. Thus $|\sigma_p(x, y, \lambda)| \leqslant e^{-y}$.
- If $x > p + \lambda^{-1}$ then for all $0 \leqslant i \leqslant p - 1$, $|1 - \sigma_1(x - i, y + \ln p, \lambda)| \leqslant \frac{e^{-y}}{p}$. Thus $|p - \sigma_p(x, y, \lambda)| \leqslant e^{-y}$.
- If $d(x, \mathbb{N}) \geqslant \lambda^{-1}$ then for all $0 \leqslant i \leqslant p-1$, $|\text{int}_1(x-i) - \sigma_1(x-i, y+\ln p, \lambda)| \leqslant \frac{e^{-y}}{p}$. Thus $|\text{int}_p(x) - 1\sigma_p(x, y, \lambda)| \leqslant e^{-y}$ because $\text{int}_p(x) = \sum_{i=0}^{p-1} \text{int}_1(x - i)$.
- Otherwise, there exists $0 \leqslant i \leqslant p-1$ s.t $|p - x| \leqslant \lambda^{-1}$. Then for all $j \neq i$, $|\text{int}_1(x-j) - \sigma_1(x-j, y+\ln p, \lambda)| \leqslant \frac{e^{-y}}{p}$ and $|\text{int}_1(x - i) - \sigma_1(x - i, y + \ln p, \lambda)| \leqslant 1/2$. Thus $|\text{int}_p(x) - 1\sigma_p(x, y, \lambda)| \leqslant 1/2 + e^{-y}$.

Finally, $\sigma_p \in \text{GSPACE}\left(\mathbb{R}^3, p\right)$ follows from Corollary 1 by applying $p - 1$ times Lemma 2.

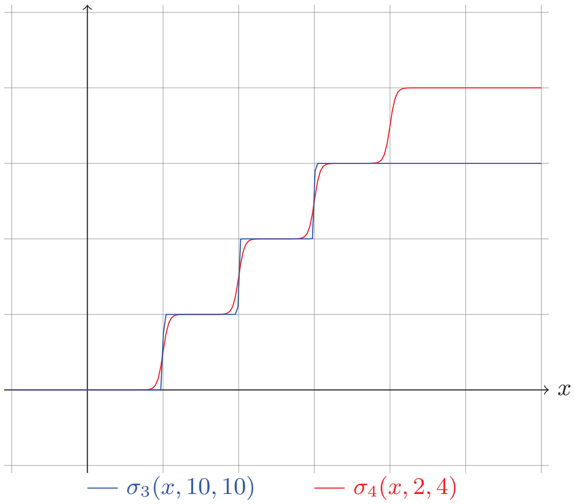


Fig. 3. Graphical representation of σ_p

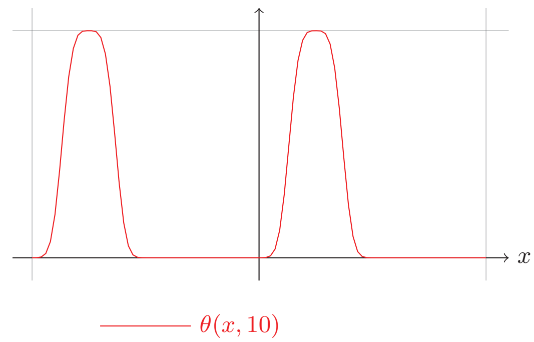


Fig. 4. Graphical representation of θ

F Proof of Lemma 8

Proof. It is trivial to see that θ is positive and 1-periodic. Furthermore, if $t \in [1/2, 1]$, then $2\pi t \in [\pi, 2\pi]$, thus $\sin(2\pi t) \leqslant 0$, so finally $-\lambda(1 - sin(2\pi t))^2 \leqslant -\lambda$ which gives the result.

Now fix $\lambda > 0$. It is easily seen that $\int_0^{\frac{1}{2}} \theta(t, \lambda)dt = 2\int_0^{\frac{1}{4}} \theta(t, \lambda)dt$ so we can restrict our analysis to $t \in [0, 1/4]$. Fix $1 > \alpha > 0$ and set $I = \{t \in [0, 1/4] \,|\, \theta(t, \lambda) \geqslant \alpha\}$. Since $\theta(\cdot, \lambda)$ is a non-decreasing function on $[0, 1/4]$ and $\theta(0, \lambda) = e^{-\lambda}$ and $\theta(1/4, \lambda) = 1$, I is of the form $I = [1/4 - x, 1/4]$. We are now looking for a lower bound on x

$$\begin{aligned}\theta(t, \lambda) \geqslant \alpha &\Leftrightarrow (1 - \sin(2\pi t))^2 \leqslant \frac{\ln \alpha}{-\lambda} \\ &\Leftrightarrow 1 - \sin(2\pi t) \leqslant \sqrt{\frac{\ln \alpha}{-\lambda}}\end{aligned}$$

Since $t \in [0, 1/4]$, $0 \leqslant \sin(2\pi t) \leqslant 1$. Thus $\arcsin(\sin(2\pi t)) = 2\pi t$ and arcsin is an increasing function

$$\Leftrightarrow 2\pi t \geqslant \arcsin\left(1 - \sqrt{\frac{\ln \alpha}{-\lambda}}\right)$$

Yet $\forall x \geqslant 0, \arcsin(1 - x) \leqslant \frac{\pi}{2} - \sqrt{2x}$

$$\begin{aligned}&\Leftarrow 2\pi t \geqslant \frac{\pi}{2} - \sqrt{2\sqrt{\frac{\ln \alpha}{-\lambda}}} \\ &\Leftarrow t \geqslant \frac{1}{4} - \frac{1}{2\pi}\sqrt{2\sqrt{\frac{\ln \alpha}{-\lambda}}}\end{aligned}$$

Thus

$$\int_0^{\frac{1}{4}} \theta(t, \lambda)dt \geqslant \underbrace{\frac{\alpha}{2\pi}\sqrt{2\sqrt{\frac{\ln \alpha}{-\lambda}}}}_{=h(\alpha)}$$

We are looking for the maximum of h over $[0,1]$ so we analyze the derivative of h

$$h'(\alpha) = \frac{1}{\sqrt{2\sqrt{\lambda^{-1}}\pi}} \left((\ln \alpha)^{1/4} + \frac{1}{4}(\ln \alpha)^{-3/4} \right)$$

which yields

$$h'(\alpha) = 0 \Leftrightarrow \alpha = e^{-1/4} \in [0,1]$$

Therefore

$$\int_0^{\frac{1}{4}} \theta(t,\lambda)dt \geqslant h(e^{-1/4}) = \frac{(e\lambda)^{-1/4}}{2\pi}$$

Finally, it easily seen that $(t \mapsto (1-\sin(2\pi t))^2) \in \mathrm{GSPACE}(\mathbb{R},1)$ by Lemma 5, Lemma 4 and Lemma 2. Then, one shows that $((t,\lambda) \mapsto -\lambda(1-\sin(2\pi t))^2) \in \mathrm{GSPACE}(\mathbb{R}\times\mathbb{R}_+, (t,\lambda)\mapsto \max(1,\lambda))$. And finally, compose with exp to get the result.

G Proof of Lemma 9

Proof. We prove it by induction on n. If $n=1$, the result is trivial. If $n \geqslant 1$, one can use the induction hypothesis and write:

$$\left|\prod_{i=1}^{n+1} x_i - \prod_{i=1}^{n+1} y_i\right| = \left|(x_{n+1}-y_{n+1})\prod_{i=1}^{n} x_i + y_{n+1}\left(\prod_{i=1}^{n} x_i - \prod_{i=1}^{n} y_i\right)\right| \leqslant |x_{n+1}-y_{n+1}|K^n + KK^{n-1}\sum_{i=1}^{n}|x_i-y_i|$$

H Proof of Lemma 10

Proof. Let $x, z \in [-K,K]^d$, let $\delta = \min_{x\neq x'\in G} \min_{i=1}^{d} |x_i - x'_i|$, $F = \max_{x\in G}|f(x)|$ and $M = K + \max_{x\in G}\|x\|_\infty$. Then using Lemma 9:

$$\begin{aligned}
|L_f(x) - L_f(z)| &\leqslant \sum_{\bar{x}\in G} |f(\bar{x})| \left| \prod_{i=1}^{d} \prod_{\substack{y\in G\\ y\neq\bar{x}}} \frac{x_i - y_i}{\bar{x}_i - y_i} - \prod_{i=1}^{d} \prod_{\substack{y\in G\\ y\neq\bar{x}}} \frac{z_i - y_i}{\bar{x}_i - y_i} \right| \\
&\leqslant \sum_{\bar{x}\in G} |f(\bar{x})| \left(\frac{M}{\delta}\right)^{d(|G|-1)-1} \sum_{i=1}^{d} \sum_{\substack{y\in G\\ y\neq\bar{x}}} \left|\frac{x_i - z_i}{\bar{x}_i - y_i}\right| \\
&\leqslant |G|F\left(\frac{M}{\delta}\right)^{d(|G|-1)-1} d(|G|-1)\frac{\|x-z\|_\infty}{\delta}
\end{aligned}$$

And using trivial inequalities:

$$\begin{aligned}
|L_f(x)| &\leqslant \sum_{\bar{x}\in G} |f(\bar{x})| \prod_{i=1}^{d} \prod_{\substack{y\in G\\ y\neq\bar{x}}} \left|\frac{x_i - y_i}{\bar{x}_i - y_i}\right| \\
&\leqslant \sum_{\bar{x}\in G} F \prod_{i=1}^{d} \prod_{\substack{y\in G\\ y\neq\bar{x}}} \frac{M}{\delta} \\
&\leqslant |G|F\left(\frac{M}{\delta}\right)^{d(|G|-1)}
\end{aligned}$$

I Proof of Lemma 11

Proof. Without loss of generality we prove it only for x. Since we took $\Sigma = [\![0, k-2]\!]$, $\forall i \in \mathbb{N}, 0 \leqslant x_i \leqslant k-2$, which gives the result because:

$$0 \leqslant \sum_{i=0}^{n} 0k^{-i-1} \leqslant 0.x = \sum_{i=0}^{n} x_i k^{-i-1} \leqslant \sum_{i=0}^{n} (k-2)k^{-i-1} \leqslant \frac{k-1}{k}$$

J Proof of Lemma 13

Proof. The constants A_3 and B_3 comes from Lemma 10 applied to δ_3 since Q and Σ are finite thus bounded and $(\bar{q}, \bar{s})$ is close to (q, s) thus bounded too. Regrouping terms and applying a gross majoration, plus the fact that $q \in Q$ and $s \in \Sigma$, we obtain $L_{\delta_3}(q, s) \in \{0, 1\}$.

$$\begin{aligned}
|\operatorname{choose}[a, b, q, s] - \operatorname{choose}[\bar{a}, \bar{b}, \bar{q}, \bar{s}]| &= |(a-\bar{a})(1 - L_{\delta_3}(q, s)) + (\bar{b} - \bar{a})(L_{\delta_3}(q, s) - L_{\delta_3}(\bar{q}, \bar{s})) + (b - \bar{b})L_{\delta_3}(q, s)| \\
&\leqslant |a - \bar{a}||1 - L_{\delta_3}(q, s)| + |\bar{b} - \bar{a}||L_{\delta_3}(q, s) - L_{\delta_3}(\bar{q}, \bar{s})| + |b - \bar{b}||L_{\delta_3}(q, s)| \\
&\leqslant \max(|a - \bar{a}|, |b - \bar{b}|) + 2MA_3\|(q, s) - (\bar{q}, \bar{s})\|_\infty \\
&\leqslant \left\|(a, b) - (\bar{a}, \bar{b})\right\|_\infty + 2MA_3\|(q, s) - (\bar{q}, \bar{s})\|_\infty
\end{aligned}$$